Journal of Science and Technology ***49****, Number 5 (2011) 119-131*

A METAMODEL FOR ANALYSIS AND DESIGN OF NON-FUNCTIONAL REQUIREMENT

Yi Liu[1,2], Zhiyi Ma[1,2], Hui Liu[1,3] Lai Wei[1,2], Weizhong Shao[1,2]

[1]*Key Laboratory of High Confidence Software Technologies, Ministry of Education*

[2]*School of Electronics Engineering and Computer Science, Peking University*
[3]*School of Computer Science and Technology, Beijing Institute of Technology*
Beijing, P. R.China

Received September 30, 2011

ABSTRACT

Non-Functional Requirements (NFRs) is critical for the development of user-satisfied software. Consequently, quite a few approaches have been proposed to facilitate NFRs' analysis and design. Some of them are general, and others are NFR-specific. However, to the best of our knowledge, there is no commonly accepted taxonomy of NFRs analysis and design. Different approaches describe the same concepts or terms in different ways. This concept diversity hinders the analysis and realization of NFRs, and interferes effective communication among stakeholders. To this end, we propose a meta model to explicate the concepts relevant to NFR analysis and design, which might provide a common foundation for these tasks. In order to validate its completeness and usefulness, terms and concepts provided in this meta model are compared with the concepts appearing in the typical analysis and design approaches. Finally, a modeling tool built upon this meta model is also developed to facilitate NFR modeling and analysis.

Keywords. Non-functional requirement; meta model; Analysis and Design

1. INTRODUCTION

In recent years, there has been an increased awareness of the importance of nonfunctional requirements (NFRs) for developing user-satisfied software. Different NFRs often make much difference on the way of implementing the software, such as the whole architecture of the software, the lower-level design model, the algorithm of a specific functionality, the deployment strategy, etc. Therefore, besides functional requirements, it is also necessary to consider non-functional requirements from the beginning of software development life cycle. In these approaches, non-functional requirements are taken as the first-order requirements, and are

analyzed and designed together with the functional requirements. The initial high-level NFRs are identified and gradually refined to lower, more-precise level of detail. Because NFRs are not operational, appropriate solutions (operationalizations) are selected to make NFRs operational and integrated into design models.

To date, approaches have been proposed to analyze and design non-functional requirements [1, 2, 3]. However, the exact collection of desirable non-functional properties and the mechanisms by which they are specified and decomposed vary among different approaches. Moreover, formulations of the concepts, notations, activities, and procedures in these approaches are diverse and different. When software developers select a method to analyze and design their non-functional requirements, it may take much of their time to identify and understand the concepts and activities in the process. Especially, when different non-functional requirements co-exist, the various concepts may make developers confused and hinder systematically handling of the requirements. Therefore, a common foundation and a clear understanding of the concepts for NFR analysis and design are valuable.

Metamodeling is the construction of a collection of "concepts" (things) within a certain domain, the obtained meta model is applicable and useful for modeling a predefined class of problems. Taking NFR analysis and design engineering as the "domain", this paper proposes a meta model to clarify the essential concepts in NFR analysis and design process. This meta model is proposed mainly by reviewing the typical NFR analysis and design approaches, such as the general approaches like the NFR framework [4], i*/Tropos approach [5], the extended problem frame [6], and the NFR-specific approaches like SQUARE approach for security[8], the DFR approach for reliability [6], the SPE approach for performance [10], etc. It mainly consists of two parts, one is the NFR core meta model which focuses on the description and analysis of NFR, the other one is the NFR tactic model which focuses on the means identified for operationalizing the NFRs. Moreover, based on this meta model, a modeling tool is developed to assist the description and modeling of NFRs.

The rest of this paper is organized as follows. Section 2 gives a brief review of the existing approaches and explains why a common foundation for NFR is needed. Section 3 elaborates the meta model. Section 4 introduces the meta model-guided activities that may be done when NFRs are analyzed and designed. Section 5 evaluates the proposed meta model and presents a NFR modeling tool. Section 6 introduces related works and section 7 concludes this paper.

2. REVIEW OF EXISTING NFR ANALYSIS AND DESIGN APPROACHES

Many approaches have been proposed to analyze and design NFRs for the aim of transition them into architecture design models [2, 3]. In this section, we first briefly introduce some typical approaches, analyzing the concept diversity existing in the approaches. Then, in section 2.2, we analyze the challenges brought by the diversity.

2.1. Prosperity of NFR Approaches

The NFR framework proposed by Chung et al. [4] is one of the typical approaches; it treats NFRs as *softgoals* to be addressed during the development process. The related functional elements are considered as *functional topics* of NFR softgoals and the design decisions are named as *operationalizations. Decomposition* and *Contribution* relationships are proposed to specify the relationships among NFRs and operationalizations. *Softgoal Independency graphs*

are provided to explicitly model the NFRs, *operationalizations* and the relationships between them. This helps better understand the NFRs refinement path and the impact of every design decision.

The i*/Tropos approach [5] inherited the concept of *softgoal* from the NFR framework. In this approach, the design decisions related to certain NFRs are taken as new *tasks* or *resources*, and *means-end* relationships are used to show how choices (tasks) are related to different softgoals, where the former basically refers the same thing with *operationalizations* in the NFR framework, and the latter is nearly the same with *operationalizations decomposition.*

Besides above approaches, the idea of problem frames [6] is also used to help the analysis of NFRs. A *problem* often refers to an undesirable situation that makes it difficult to achieve a goal. *Threats, vulnerabilities et al* are identified so as to thoroughly understand NFRs and the running environment.

Misuse cases [7] are other approaches, based on UML, to handle NFRs. The term *misuse case* has derived from use case. It treats NFRs as *quality goals*, and uses *countermeasures* to describe the NFR-related decisions. They believe that the analysis of NFRs in terms of *assets, threats, misuses* and *countermeasures* helps to complement software requirements.

Besides the general approaches, there are also some approaches proposed to handle specific NFR or NFRs. The SQUARE methodology [8] is used to help organizations build *security* into the early stages of the product life cycle. It provides means for eliciting, categorizing and prioritizing *security requirements* for information technology systems and applications. B*usiness* and *security goals* are outlined and analyzed along with the artifacts and documentation of the relevant system. The security related *risks* and *threats* are identified, and a structured *risk assessment* is executed to determine the likelihood and impact of possible threats to the system .Moreover, two subsequent stages are spent categorizing and prioritizing these requirements, and *security means* are selected to satisfy the requirements.

Design for reliability (DFR) [9] is an emerging discipline that refers to the process of designing *reliability* into products. During system design, the top-level r*eliability requirements* are then allocated to subsystems, *failure*, *fault* or *errors* that may affect the reliability is identified, and *design principles* are proposed to realize the reliability requirements.

Software Performance Engineering (SPE) [10] is a systematic approach to the cost-effective development of software systems to meet *performance goals*. It identifies *risks*, uses quantitative methods to identify satisfactory designs support the *performance solutions* and to eliminate those that are likely to have unacceptable performance before developers invest significant time in implementation.

2.2. Challenges

As discussed above, various concepts and activities were proposed to treat the analysis and design of non-functional requirements. This concept diversity hinders the analysis and realization of NFRs, and interferes effective communication among stakeholders. For example, when developers try to select a method to analyze and design NFRs, it will take much of their time to identify and understand the concepts and activities. Therefore, a common foundation and a clear understanding of the concepts for NFR analysis and design is necessary. meta modeling is the process of construction of a collection of concepts (things, terms, etc.) within a certain domain. It is the analysis, construction and development of the frames, rules, constraints, models

and theories applicable and useful for modeling a predefined class of problems. Taking the NFR analysis and design approaches as the research domain, we propose a NFR meta model to clarify the concepts, terms and relations emerged in these approaches, in order to provide a common foundation for understanding NFRs and these approaches.

3. THE PROPOSED META MODEL

In this paper, the NFR meta model is developed based on commonality analysis and generalization from related literatures in Requirement Engineering and Software Engineering communities. An important core of the meta model is about the terms and concepts indispensable to NFR modeling and analysis. Because the aims include facilitating the transition from requirement to design, some design related concepts are also addressed. The meta model contains many concepts, to cope with the complexity and facilitate the understanding, we organize it into two packages. The first one is the *NFR core meta model*, which mainly focuses on the nature of NFRs and the concepts exist in NFR analysis process, the second one is the *NFR Tactic meta model*, which concerns with the concepts and relations about identifying and selecting operationalizations (design tactics) for designing the NFRs.

3.1. The NFR Core meta model

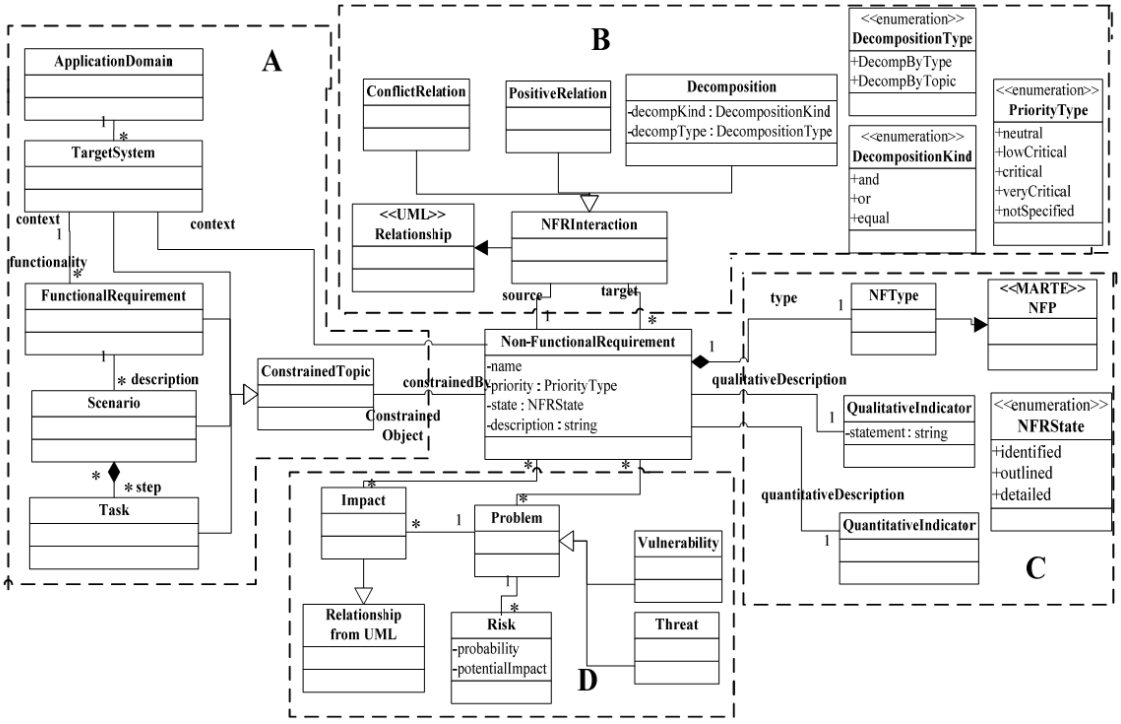

Figure 1. The NFR core metamode

The *NFR* Core meta model (Figure 1) shows the concepts frequently used in the process of NFR analysis, including the basic description of NFR, the relationships between the NFR and the functional part, the domain problems that will affect the NFRs, and interdependencies between different NFRs.

1) Basic description of NFRs

The term "non-functional requirement" (NFR) is used to delineate requirements focusing on "how good" software does something as opposed to the functional requirements (FRs), which focus on "what" the software does [11]. The element *TargetSystem* in this meta model refers to the software system under development; its aim is to resolve certain problems in the business domain represented by the element *ApplicationDomain.* FRs and NFRs are originated from the domain and formed according to the target system. First, we distinguish a concrete *NFR* from the nonfunctional property (NFP) it belongs to. For example, the NFR "The database of our new system shall handle 1000 queries per second." belongs to the NFP "performance" or more specifically the NFP "work load of database". In our meta model, we use *NFType* to represent the non-functional property specified by the NFR (Part C in Figure 1). It is extended from the element *NFP* from the MARTE profile [12]. Besides the type, indicators/values are also needed to specify an NFR, either quantitatively or qualitatively. Thus, two model elements respectively named as *QuantitativeIndicator* and *QualitativeIndicator* are provided to represent them, e.g., "1000 queries per second" is a quantitative indicator of the NFR *performance*, while the qualitative indicator is generally a value from a list of allowed values.

Moreover, since NFRs can be seen as requirements that constrain or set some quality attributes upon functionalities, they usually have close relationships with the functional part of the target system. That is, when we want to understand a NFR, we also need to know the object constrained by it or responsible for its satisfaction. E.g., for the NFR "Transfer fund in the internet bank must have high security", "transfer fund" is the functionality that constrained by the NFR *security* and is responsible to make the NFR satisfied. Part A of Figure 1 shows the relationships between NFRs and the functional parts. *ConstrainedTopic* is used to specify this object, the *TargetSystem*, *FRs*, and the artifacts derived from the *FRs*, such as the *Task*s refined from the FRs and the *ModelElement* used to specify the design and implementation of the tasks, could be taken as a *ConstrainedTopic*. This means that these artifacts or objects are constrained by the NFR and are responsible for the satisfaction of the NFRs.

2) Problems related to NFRs

In order to better understand an NFR, we sometimes need to understand the problems in the application domain that may affect them. E.g., *network latency* may be a challenge problem for a *performance* requirement. Part D of Figure 1 shows the general concepts existing in problem analysis of an NFR. A *problem* in this meta model refers to an undesirable situation that makes it difficult to achieve the NFR. *Vulnerability* and *Threat* are two typical kinds of problem. A *vulnerability* is a weakness or flaw in a system which is likely to bring negative impact on certain NFR. The term *threat* refers to the source and means of a particular type of attack or negative impact. It describes a potential cause of an incident, which may result in harm of systems and organization. This term is often used in security engineering. Moreover, a vulnerability of the system may be exploited by one or more threats. The term *impact* is used to represent the relationship between a *problem* and its related *NFRs*. The *risk* refers to the likelihood of being targeted by a problem. A risk assessment is usually performed to determine the most important potential NFR breaches to address now, rather than later.

3) Interdependencies between NFRs

Generally, NFRs in a target system are not always independent with each other. Part B of Figure 1 shows the interactive relationships among NFRs. The element *NFRInteraction* is used to represent the relationships, which is specified into three subtypes, *Decomposition*, *ConflictRelation* and *PositiveRelation*.

The *Decomposition* relationship is used to represent the relationship between a high- level NFR and more specific sub-NFRs, e.g., a *security* requirement may be decomposed into *confidentiality* or *integrity*. In each decomposition, the offspring NFRs can contribute partially or fully towards satisfying the parent which is specified by the attribute *decompKind* (which value is listed in the enumeration *DecompositionKind),* and each decompoistion can be carried either across the *type* dimension or the *constrainedTopic* dimension (which is specified by the attribute *decompType).* Additionally, the attribute *state* of a *NFR* is used to indicate the degree of the decomposition, such as *identified, outlined,* or *detailed*.

The relationships *ConflictRelation* and *PositiveRelation* refer to the fact that the achievement of one NFR at some extent may hinder or help the achievement of other NFR, e.g. the *security* requirement and *performance* requirement at the functionality *transferring money* from internet bank are interrelated with each other. Therefore, when different NFRs coexist in a software system, priorities of them may need to be identified. In this meta model, we add the attribute *priority* to an NFR to specify the priorites, and in the enumeration *PriorityType* , we give some possible values of the attribute.

3.2. The NFR Tactic meta model

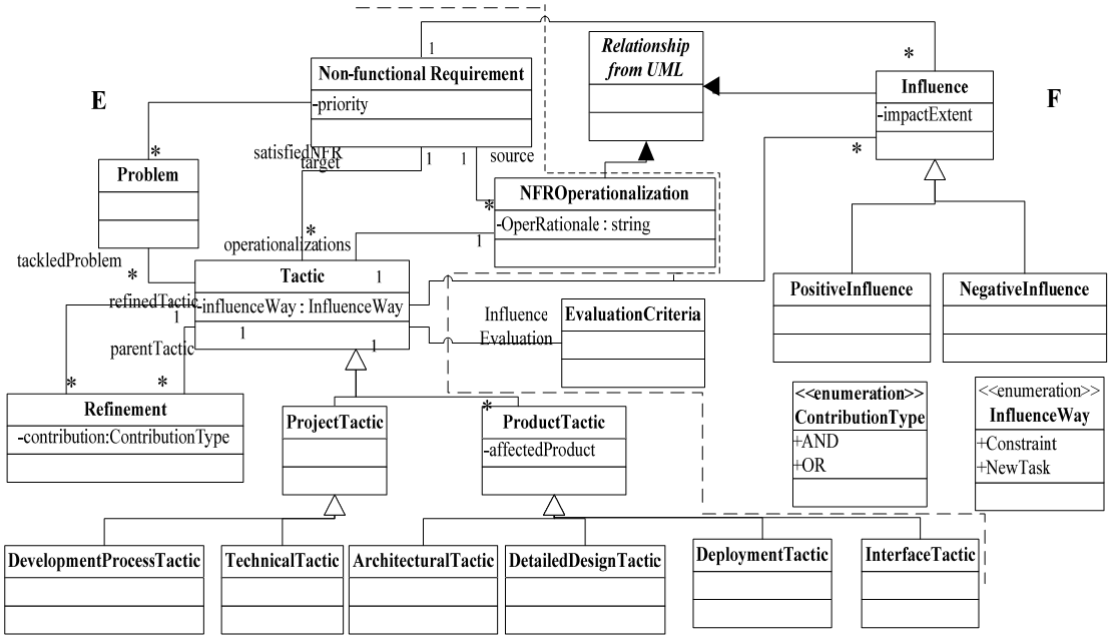

Figure 2. The NFR Tactic metamode

When NFRs are decomposed enough, the next thing to do is operationalizing them into solutions so as to make them satisfied. We define the term *Tactic* to represent the solutions, and the inferred taxonomy of tactic is presented in Figure 2. The left part (part E) mainly shows the concepts about the tactic and its satisfied NFR, the right part (part F) shows the interdependencies between the tactics and other *NFRs*.

1) The design tactics of NFRs

A *NFR Tactic* is a fine-grained reusable scheme that provides a solution built from experience to help to achieve a non-functional requirement. E.g., *password authentication* is a frequently-used tactic for the NFR *security*. It might be operations, functions, data representations and architecture design decisions (e.g., architecture pattern, design pattern) in the system to meet the needs stated in the NFRs. The association-end attribute *satisfiedNFR* indicates the NFR that the tactic aiming to achieve, and also an optional attribute named *tackledProblem* is provided to present the *problem* that the tactic solved. Moreover, one tactic could be refined to other tactics; this is denoted by the relationship *refinement.* Similar to NFR *Decomposition*, a tactic can be *ANDed or ORed*. To explicitly represent the relationships between NFRs and corresponding tactics, we also provide a model element named *NFROperationalization.* It extends the *Relationship* element defined in UML, the related elements are *non-functional requirement* and *tactic*. Moreover, an attribute *operRationale* is also defined to specify the reason for the operationalization.

As viewed from the object the tactics affected, we classify the tactics into two types: *ProjectTactic* and *ProductTactic*. The term *ProjectTactic* refers to the tactics applied to the software project, such as the tactics placed on the development process which referred to as *DevelopmentProcessTactic*, the tactics placed on the development technique which was referred as *TechnicalTactic* in this meta model, etc. In addition, the term *productTactic* refers to the tactics applied to the artifacts obtained in the development process, such as the tactics put into the software architecture, the tactics put into the detailed design, the deployment or some interfaces. These tactics are respectively represented by *ArchitecturalTactic*, *DetailedDesignTactic*, *DeploymentTactic* and *InterfaceTactic*. Furthermore, the way the tactics influences the software development could be divided into two types, one is placing constraint on the affected objects, and the other is adding new task (also can be seen as new functionality); this is represented by the attribute *influenceWay* of the tactic.

2) Interdependencies between tactics and NFRs

Since the solution of an NFR may hinder or help the achievement of other NFRs, before selecting solutions of an NFR, it is necessary to evaluate the influences and make the interdependencies explicit. Part F of Figure 2 shows the relevant concepts. The element *EvaluationCriteria* is used to specify the criteria used to judge the tactic's impact on other NFRs. The term *Influence* is proposed to represent the relationships between the tactics and the NFRs. It is pointed from a tactic to an NFR. Moreover, we specialize the influence into *PositiveInfluence* and *NegativeInfluence*. The former means that the tactic selected to achieve one NFR will also help the achievement of the other NFR; the latter means that the tactic selected for one NFR will hinder the achievement of the other NFR. For example, a tactic selected for *confidentiality* may hinder the achievement of *performance* requirement, but may help the achievement of *accuracy* requirement. The influence evaluation will help developer balance the tradeoffs and make rational decision. In addition, an attribute *impactExtent* is also provided in this meta model, users could further specify the extent of impact that a tactic have on other NFR.

4. META MODEL-GUIDED ANALYSIS AND DESIGN ACTIVITIES

In section 3, we propose a meta model to elaborate the general concepts relevant to NFR analysis and design. We illustrate here about how the meta model will help the process of NFR analysis and design.

Unlike FRs which could be directly mapped into design model, the initial descriptions of NFRs are often abstract and just some user-required properties; which usually means that they are not operational in the beginning. Therefore, more factors should be taken into consideration in the analysis and design of NFRs and more tasks need to be done. Based on the proposed meta model, we could reason out following tasks that may need to be done when analyze and design the NFRs.

1) Identifying of NFRs from different viewpoints and different levels of detail.

- *Individual NFR description*

First, from part A and part C of the NFR core meta model, we could know that: a) the identification of NFRs should consider both the target system and the application domain, b) when describe an NFR, we should specify its type *(NFtype)*, i.e., the required non-functional property (NFP), and the constrained topic first; c) when we identify the topics constrained by NFRs, we could consider the whole system, one certain functional requirement, a scenario or even more detail tasks; c) the value of the NFPs specified in the NFRs could be qualitative or quantitative;

- *NFR refinement and negotiation*

Besides above basic NFR description described in part A and part C, from part B we can see that there are different interdependencies between NFRs. First, abstract high-level NFRs could be decomposed into low-level concrete NFRs either by its type or the constrained topic using AND, OR or Equal operator. This will help a detailed understanding of the NFRs. Second, there are positive or negative Contribution relationships between the NFRs that have no parent/child relationship, these relationships may need to explicitly identified to help the trade-offs.

- *Problems analysis*

Moreover, from part C we could know that when we analyze NFRs, we could still identify related problems, such as certain threat or vulnerability. Since a problem often refers to an undesirable situation that makes it difficult to achieve a goal, the identification of problems will help a more detailed analysis of NFRs.

2) Identification and Evaluation of NFR Solutions

A major step in the understanding of NFR is the distinction between the NFR and means to achieving it [13]. The division of NFR meta model into *NFRCore* and *NFRTactic* fits well with this viewpoint. Based on the *NFRCore* meta model, we could identify the key concepts and entities in NFR analysis process. Then, following the *NFRTactic* meta model, we could derive tasks that may be done when design NFRs.

- *Identification of possible means*

From part E of the NFR tactic meta model, we could see that: a) To satisfy an NFR, possible tactics should be identified and associated to the NFRs by the *NFROperationalization* relationship; b) the tactics may be proposed to solve certain problems; c)the type of tactic is

various, and one abstract tactic could be refined into more concrete ones.

- *Evaluate and select the means*

Moreover, to select appropriate tactics for each NFR, an evaluation procedure should be applied. From part F of the NFR tactic meta model, we could see that, besides the satisfied NFR, a tactic may have positive or negative influence on other NFRs. These relationships should also be identified to help the selection.

5. EVALUATION AND TOOL SUPPORT

This paper presents, through a meta model, glossaries and taxonomies for NFR analysis and design. In this section, we evaluate the meta model by comparing it against the concepts employed by existing NFR approaches. We also introduce a NFR modeling tool developed based on this meta model.

Table 1. Comparison of the concepts between the proposed metamodel and typical approaches

Concepts in Existing common methods / Concepts in the proposed Metamodel	General Method			Domain Specific Method		
	The NFR framework [4]	i*/Tropos[5]	Misuse-oriented method [7]	Software Reliability [9]	Software Performance Engineering[10]	Security Engineering/SQUARE [8]
Non-Functional Requirement	Softgoal	Softgoal	QualityGoal	Reliability goal	Performance goal	Security goal
ConstrainedTopic	FunctionalTopic	-	Asset	-	*implicit*	-
NFType	Type	-	QA	Reliability	Performance	Security
Priority	Critical	-	-	-	-	Priority
Tactic	Operationalization softgoal	Task	CounterMeasure	Design principles	Performance Solutions	Security means
OperRationale	Claim softgoal	Belief	-	-	-	-
NFRDecomposition	Decomposition	Decomposition	-	-	-	-
NFROperationalization	Satificing	Means-ends	-	-	-	-
Influence/positiveInfluence/negativeInfluence	make, help, break, hurt	Contribution	-	-	-	-
Problem	-	-	-	-	-	Sources of insecurity
Risk	-	-	-	-	Risk	Risk
Threat	-	-	Threat	Failure, Fault, Error	-	Threat
Vulnerability	-	-	-	-	-	-
QuantitativeIndicator	-	-	implicit	Reliability properties	Performance properties	Risk properties
Senario	-	-	BusinessGoal -	Operational profile	Key Scenarios/ Execution Context	-

5.1. Concept Evaluation

The evaluation criterion for the discussed meta model is that the common foundation for NFRs (i) should be generally acceptable for stakeholders in RE Community, (ii) could help people quickly understand the concepts and relationships in NFR analysis and design, accurate

and consistent [17].

"Generally accepted" means that the knowledge and practices described are applicable to most projects most of the time, and the accuracy and consistency of the meta model also rely upon its application by the research community. In this paper, we demonstrate these two points through comparison the concepts in our proposed meta model and those in the common NFR analysis and design approaches. Table I shows the corresponding relationships of the concepts in the comparison.

The leftmost column shows the concepts proposed in our meta model, the other columns show related concepts emerged in the typical NFR analysis and design approaches. They often have the same meaning or indicate the same kind of object with the leftmost concept. The "-" in the table means that there is no corresponding concept in the method or the concept has not been explicitly identified in the method. From the table, we could see that nearly all the concepts in the typical approaches have corresponding concept in our meta model, and comparing to each approach, our meta model cover a broader concept space. For the second criteria, since meta modeling is a good means to analyze, construct and develop a collection of concepts (things, terms, etc.) within a certain domain, we believe the proposed meta model could help people understand NFR analysis and design related concepts more easily.

5.2. Tool Support

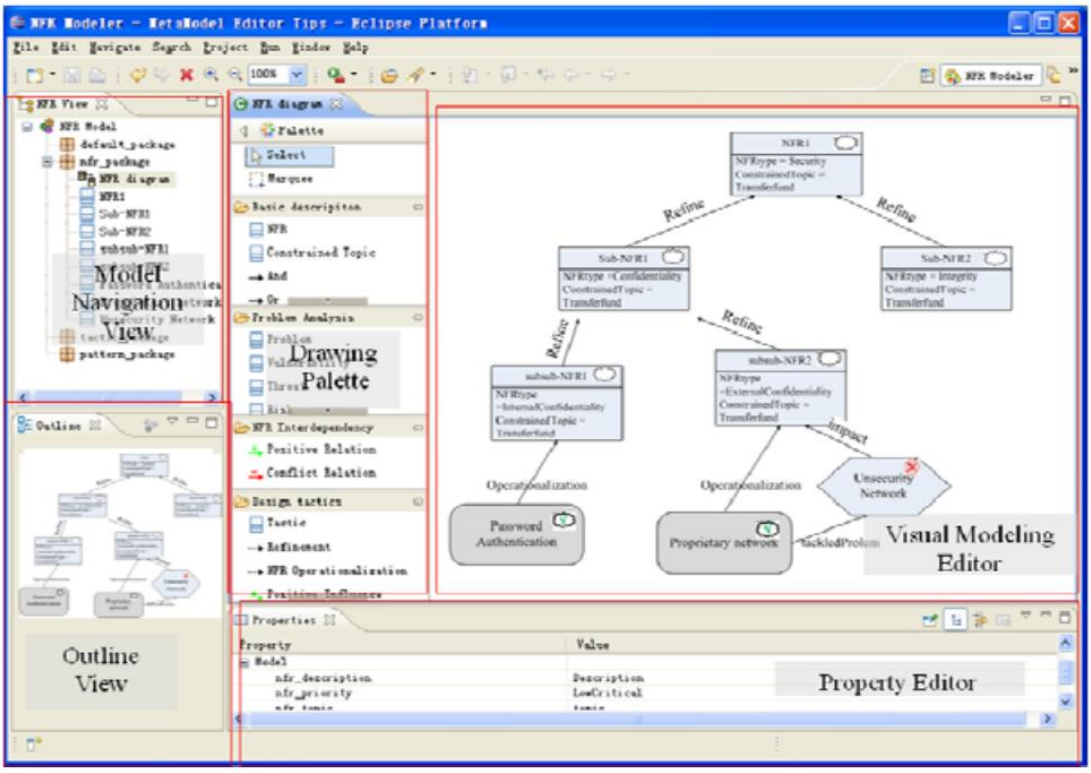

Figure 3. The NFR Modeling tool

The assistant of a modeling tool is important for successful NFR analysis and design. Tool support also plays a critical role in unifying modeling concepts. Furthermore, tool support is indispensible for meta model evaluation on real projects. To this end, with the proposed meta model, we have developed a modeling tool to model NFRs and the relevant objects. Figure 3 shows a snapshot of the tool. In the drawing palette, elements are divided into different categories to facilitate the understanding and usage of the tool.

Furthermore, the meta model proposed in this paper is constructed mainly by detailed literature analysis and our development experience of some projects. Although we have evaluated its completeness by comparing it with the typical approaches, it still needs further evaluations, such as applying it to more real projects, discussing them with other RE communities, etc.

6. RELATED WORK

In the RE literature, there are relatively few proposals for conceptual models pertaining to NFRs. The existing approaches could be generally placed into two categories: quantitative ones and qualitative ones. The former ones are mainly used to specify the quantitative aspect of an NFR (i.e., describing the value of a required nonfunctional property), which could be used in the evaluation or calculation process of certain NFR. Most of the existing meta models are belong to the former ones, such as the UML profiles SPT [14], QoS&FT [15], MARTE [12], etc. They provide extensive taxonomy of NFR related concepts in UML for quantitative specification and modeling of NFRs. However, these profiles are mainly used when analyzing NFRs quantitatively. While the meta model proposed in this paper mainly focuses on the qualitative analysis and design of NFRs, i.e., enforcing NFR analysis and design into the software development process.

Towards the qualitative meta model, Supakkul et al. [16] proposed a UML profile to represent NFRs and FRs using the goal oriented NFR framework. It also presents a procedure profile. However, this profile mainly aimed at one specific analysis approach (i.e., the NFR framework), while we focus on a generic meta model independent with specific method. Kassab et al. [17] proposed an ontology based approach to NFR conceptualization. They provided three views for NFRs: intramodel dependency view, intermodal dependency view and a view that represents the measurement process and the concepts used to produce measures to measurable NFRs. While in our paper, we mainly focus on the concepts related to NFR analysis and design, we propose a *NFRCore* meta model to express the terms about NFR identification and analysis, and a *NFRTactic* meta model to show the design related concepts. Tian et al. [18] proposed a context awareness NFR meta model for network software. Domain ontology is imported to guide non-functional requirement acquisition, knowledge and rules are provided by domain ontology to induce NFRs in specified domains. This work is mainly ontology-based, and the concern is the description of the context. While in our paper, the meta model considers generally the terms and taxonomy in NFR analysis and design process, context is just one part of it.

Moreover, besides the meta models, we also summarize the basic activities involved in the process of general NFR analysis and design, which will help developers better understand an analysis and design approach or customize their own approach.

7. CONCLUSIONS AND FUTURE WORK

Analyzing Non-functional requirements and designing them is important for enforcing them into the software development process. Even though various approaches have been proposed to do this, the concepts, notations and procedures in them are diverse. It is difficult for users to describe their own NFRs accurately and precisely, and it is also time-consuming for developers to analyze and design the various NFRs systematically. In this paper, we propose a meta model to specify the terms and taxonomy relevant to NFR analysis and design, and introduce how they will guide the analysis process. A modeling tool is also developed to support the NFR analysis and design process. In the future work, we plan to continue our efforts in the NFRs meta model validation in different contexts, especially apply them into the real project, so as to further evaluate its completeness and usefulness.

REFERENCES

1. L. M. Cysneiros, et al. - Nonfunctional Requirements: From Elicitation to Conceptual Models, IEEE Transactions on Software Engineering **30** (2004) 328-350.
2. L Chung, J. do Prado Leite - On Non-Functional Requirements in Software Engineering, Conceptual Modeling: Foundations and Applications (2009) 363-379.
3. A. Matoussi, R. Laleau. - A Survey of Non-Functional Requirements in Software Development Process, Technical report.TR-LACL-2008-7.
4. L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos - Non-Functional Requirements in Software Engineering. 1st ed. Springer, 1999.
5. Y. Eric - Towards modeling and reasoning support for early-phase requirements engineering, Proc. Third IEEE Inl.Symposium on Reqirement Engineering, 1997, pp. 226-235.
6. Ian Alexander - Misuse cases help to elicit non-functional requirements, Computing & Control Engineering Journal (2009) 40-45.
7. S. Supakkul, L. Chung - Extending problem frames to deal with stakeholder problems, SAC'09, 2009.
8. Mead Nancy R., and Stehney Ted. - Security quality requirements engineering (SQUARE) methodology, SIGSOFT Softw. Eng. Notes (2005) 1-7.
9. Dana Crow, Alec Feinberg - Design for Reliability, CRC press, 2001.
10. C. U. Smith, L. G. Williams - Software Performance Engineering: A Case Study Including Performance Comparison with Design Alternatives. IEEE Transactions on Software Engineering **19** (7) (1993).
11. M. Glinz - On Non-functional Requirements, In: 5th IEEE International Conference on Requirements Engineering, (RE '07), 2007, pp. 21-26.
12. Object Management Group - UML Profile for Modeling and Analysis of Real-time and Embedded Systems version1.0. OMG Document, formal/2009-11-02, 2009.
13. B. Paech, D. Kerkow - Non-Functional Requirements Engineering - Quality is Essential,

10th Inl' Workshop on Requirement Egnineering: Foundation for Software quality, REFSQ'04.

14. Object Management Group - UML profile for schedulability, performance and time, OMG Document, formal/05-01-02, 2005.
15. Object Management Group - UML Profile for Modeling QoS and Fault Tolerance Characteristics and Mechanisms. version1.1. OMG Document, formal/08-04-05, 2008.
16. S. Supakkul, L. Chung - A UML Profile for Goal-oriented and Use Case Driven Representation of NFRs and FRs. In: SERA'05, 2005.
17. M.Kassab, et al.- An Ontology Based approach to Non-Functional Requirements Conceptualization, In: ICSEA'09, 2009.
18. Tian JingBai et al. - A context Awareness Non-functional Requirements meta model based on Domain Ontology, IEEE International Workshop on Semantic Computing and Systems, 2008.

Corresponding author:

Yi Liu,

Key Laboratory of High Confidence Software Technologies, Ministry of Education

School of Electronics Engineering and Computer Science, Peking University

Email: *liuyi07@sei.pku.edu.cn*